Cyclic redundancy codes are commonly used for error detection in networks and other applications. A CRC can be thought of as a function for a data word that can be used to detect corrupt data. Mathematically, a binary bit stream is treated as a polynomial over GF(2) (i.e., each polynomial coefficient being a zero or one) and performing polynomial division by a generator polynomial G(x), commonly called a CRC polynomial. The remainder of that division operation provides an error correction value sent as a frame check sequence (FCS) within a network message or stored as a data integrity check. This value is appended to the message stream and transmitted. The CRC computation takes the form of a bitwise convolution of a data word against a binary version of the CRC polynomial.

Error detection is performed by comparing an FCS value and the computed FCS values are not equal.

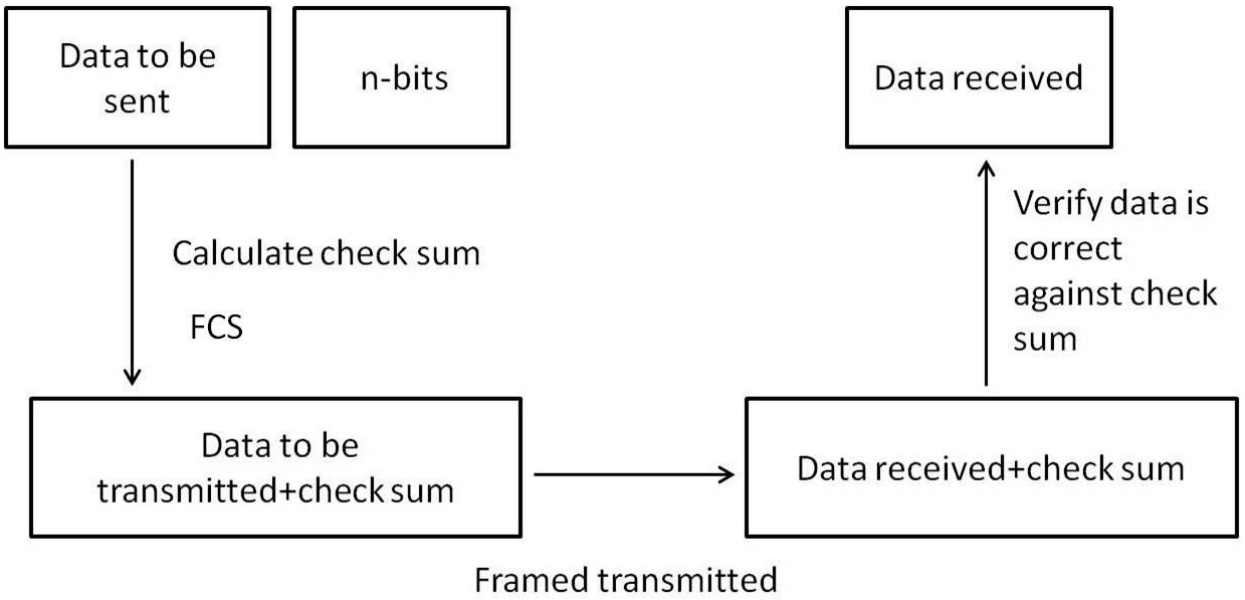

Figure 1: An illustration of the Cyclic Redundancy Check method

In our preliminary simulations, we used a generator polynomial $G(x)=x^6+x^5+x^4+x^3+1$. As per CRC protocol, 6 zeros are padded to the message and encoded to be transmitted. We use linear shift registers and XOR gates to calculate the check sum.

Transmission Gate

The transmission gate was designed to be implemented in the master slave register.

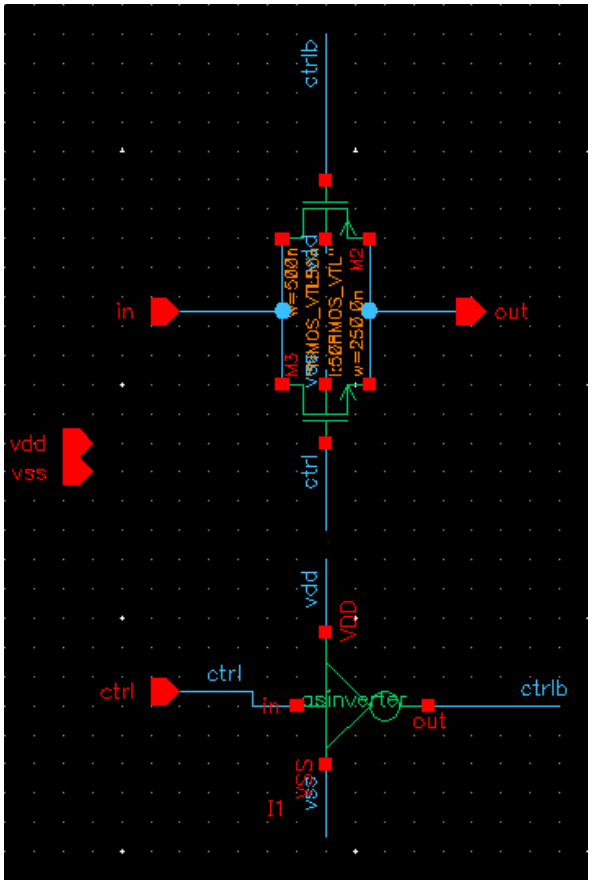

Figure 2: Transmission Gate

Positive Edge Triggered Master Slave Flip Flop

We designed a positive edge triggered master-slave register with a synchronous reset option to flush the registers to hold a 0 initially. The register was designed using transmission gates. The circuit consists of two D flip-flops connected together. When the clock is low, the D input is stored in the first latch, but the second latch cannot change state. When the clock is high, the first latch's output is stored in the second latch, but the first latch cannot change state.

The result is that output can only change state when the clock makes a transition from low to high.

XOR

The XOR gate was designed using transmission gate to follow the truth table shown below. When both the inputs are 0's or 1's, it outputs a 0. If one of the inputs is high, the output is high.

x	y	x XOR y
0	0	0
0	1	1
1	0	1
1	1	0

Figure 2: Truth table of XOR gate.

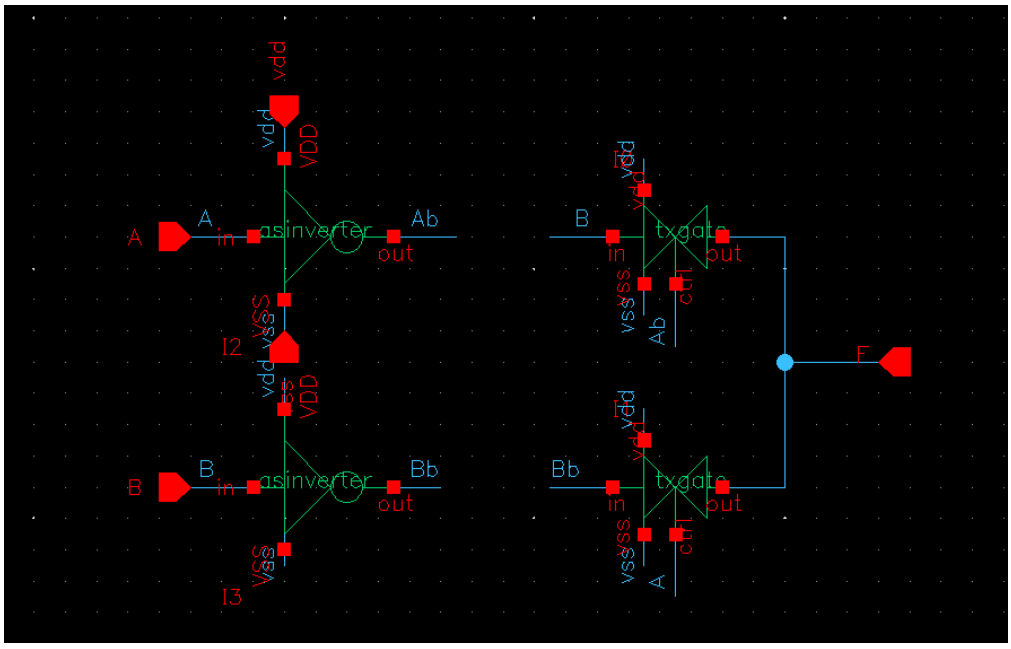

Figure 3: Positive edge triggered master-slave flip flop with reset using MUX

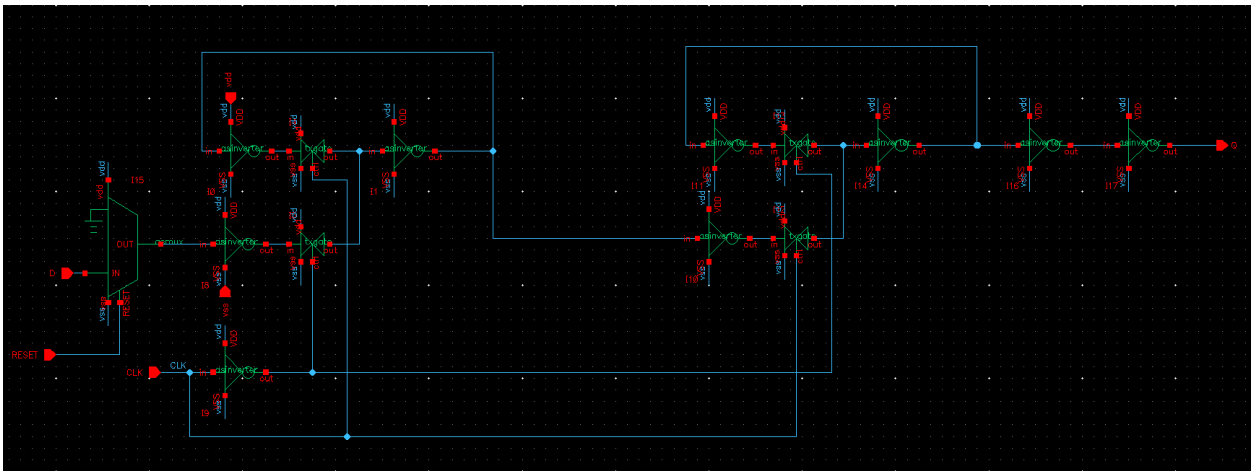

Figure 4: XOR gate using transmission gate

The overall CRC circuit is a 6-bit CRC designed to test a message bit 6-bit long. The schematic of the circuit is shown. The CRC-6 is implemented as a linear feedback shift register (LFSR). The shift register computes the LSFR function. This design requires the need for the initial values of the shift register to be initialized to zero. In this simulation, the generator polynomial, G(x) is 1111001b. After the register is initialized, at the falling edge of the clock, the most significant value (MSB) of the message bit with n appended zeros enters the shift register computing the CRC algorithm. As the number of message bits in this example is 6, the entire message enters the shift registers after 6 clock cycles. After 12 clock cycles (6bits for message+6bits for G(x)), the CRC is computed. These CRC bits are appended to the message sequence and transmitted.

The ratio of PMOS:NMOS is sized to 500nm/50nm:250nm/50nm. This is done to simulate high speed operation. To initialize the flip flops before operation, a synchronous reset was implemented using a multiplexor. To cater to the set up and hold time violations, a buffer was added to the output of the flip flops.

Simulation

At the 12th clock cycle, the CRC is obtained to be appended to the message sequence. Due to the difference in the timing between the XOR and flip flop operation before the buffers were added, glitches

were observed at the output of the circuit. The buffers reduced the glitches considerably. Going forward, to address circuit level issue, we plan on investigating the trade off between the glitches and the use of buffers. Also, we plan on looking into different register topologies for implementation. The trade off between implementing a low power CRC against high speed.

Application

CRC-16 is used in IBM's BISYNCH communication standard. The CRC-CCITT polynomial, also known as ITU-TSS, is used in communication protocols such as XMODEM, X.25, IBM's SDLC, and ISO's HDLC. CRC-32 is also known as AUTODIN-II and ITU-TSS (ITU-TSS has defined both 16- and a 32-bit polynomials). It is used in PKZip, Ethernet, AAL5 (ATM Adaptation Layer 5), FDDI (Fiber Distributed Data Interface), the IEEE-802 LAN/MAN standard, and in some DOD applications.